

Klasa Simulation, wersja 7

```
unit Simulation: PriorityQueues class;
  unit SimProcess: elemFIFO coroutine;

    var event: EventNotice;  (* zapewnij że, event.p = this SimProcess *)

    unit isIdle: function: Boolean;

    begin
      result := (event=none); (* niezaplanowany wttw gdy event = none*)
    end isIdle;

    unit isTerminated: function: Boolean;

    begin
      result := finished;
    end isTerminated;

  begin
    return; (* koniec konstruktora, konstruktor jest pusty *)
  inner; (* tu kompilator wstawi wątek klasy pochodnej *)
  finished :=true; (* zakończono wątek klasy pochodnej *)
  call passivate;
  raise Error; (* wrzucamy wyjątek gdyby ktoś wznowił, przez attach, ten współprogram *)
end SimProcess;

unit EventNotice: elemPQ class(p: SimProcess, t: time);
  unit less: virtual function(e: EventNotice): Boolean;
  begin
    result:= t < e.t
  end less;
end EventNotice;
(* własność S2 jest spełniona, ponadto obiekty tej klasy mogą być wstawiane do kolejki PQ *)

unit PlanSymulacji: QueueHead class;
  unit schedule: procedure(p:SimProcess, t: time);
  var ev: EventNotice;
  begin
    if p. terminated
    then
      raise ErrorDo_not_ScheduleTerminatedProcess;
    else
      ev := new EventNotice(p, t);
      if not p.idle
      then
        call SQS.delete(p.event);
      endif;
      call SQS.insert(ev);
    end;
  end;
```

```

        p.event:=ev;
    endif;
end schedule;
unit hold: procedure(dt: time);
begin
    call SQS.schedule(currentProcess, currentTime+dt);
    call SQS.chooseProcess;
end hold;
unit run: procedure(p: SimProcess); ...
unit passivate: procedure; ...
unit cancel: procedure; ...
unit chooseProcess: procedure;
    var e: EventNotice;
    begin
        (* wartością SQS.min jest najmniejszy element, typ elemPQ*)
        e:=SQS.min qua EventNotice; (* dzięki qua odzyskujemy poprawny typ *)
        (* zmienne currTime i currProcess są prywatnymi zmiennymi w klasie Simulation *)
        currProcess:= e.p;
        currTime := e.t;
        attach(e.p);
    end chooseProcess;
unit currentProcess: function: SimProcess;
begin
    result := currProcess;
end currentProcess;
    unit currentTime: function: time; ...
end PlanSymulacji;

unit time: class ... end time;

var SQS: PlanSymulacji;
(* własność S1 jest zagwarantowana, SQS jest kolejką priorytetową!*)
var currProcess: SimProcess;
var currTime: time;
end Simulation;

```